LEARNING FROM MULTI-SOURCE WEAK SUPERVISION FOR NEURAL TEXT CLASSIFICATION

A Dissertation
Presented to
The Academic Faculty

By

Wendi Ren

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in the
College of Computing
Department of Computational Science and Engineering

Georgia Institute of Technology

August 2020

LEARNING FROM MULTI-SOURCE WEAK SUPERVISION FOR NEURAL TEXT CLASSIFICATION

Thesis committee:

Dr. Chao Zhang
College of Computing
Georgia Institute of Technology

Dr. Tuo Zhao
H. Milton Stewart School of Industrial & Systems Engineering
Georgia Institute of Technology

Dr. Shamkant B. Navathe
College of Computing
Georgia Institute of Technology

Date approved: July 27, 2020

You can overcome anything, if and only if you love something enough.

Lionel Messi

To my family, for their endless love.

ACKNOWLEDGMENTS

First and foremost, I am thankful beyond words to my perfect advisor, Prof. Chao Zhang. It is my great fortune to work with Chao as soon as he joined Georgia Tech. He guides me with all kinds of work, no matter how big or trivial. From the coding style to paper writing, from the detailed research idea to the big picture of research impact, Chao always instructs me with passion and encouragement. He is also a life-long role model to me, who lightens me in both the research career and being a self-motivated, a hard-working, and a nice person.

Then I would like to thank my thesis committee members, Prof. Shamkant B. Navathe and Prof. Tuo Zhao. They have not only provided useful suggestions for the thesis, but also shared their valuable research experience to me, which is my precious fortune for the future career.

I am very grateful for to all my collaborates. My lab mates – Rui Feng, Yinghao Li, Lingkai Kong, Yi Rong, Hanting Su, Ruijia Wang, Yue Yu, Rongzhi Zhang, and Yuchen Zhuang – are the best teammates I have ever seen and worked with. They inspired me a lot for the research ideas, helped me with technique skills, and also cared me very much in the everyday life. Meanwhile, the cooperation with Prof. Cassie S. Mitchell's lab and her student David Kartchner is an unique and precious experience. Dr. Mitchell taught me how to face the difficulties with her special growth story. Special thanks to Yinghao Li, Hanting Su, and David who contributed a lot to this research project. It is also my great pleasure to work with Dr. Mahdi Roozbahani as his teaching assistant for three semesters. He is not only the lectures, but also a good friend to encourage and support me. My memory at Gatech is so wonderful due to their participation.

I am a luck guy to have these good friends in Gerogia Tech: Xi Cheng, Jiahao Cui, Binze Cai, Hua Huang, Junghyun Kim, Chao Li, Chen Liu, Yingjie Qian, Di Wu, Bochao Wei, Hairong Wang, Ciyuan Yu, Yuanlai Zhou, and so on. They gave me so much support

whenever I was suffering difficulties in research or in life. They also gave me research guide like how to make presentations, how to write drafts, and how to find interesting topics. I cannot image how the life will be without their love and encouragement, especially during the isolation time during the COVID-19.

Last but not the least, my deepest gratitude to my family. My parents, Xudong Ren and Liping Cao, always give me the biggest support and freedom to pursue whatever I value. My grandmother, Ruilan Liu, gives me endless love from I was born until now. My family is my forever harbor, and my lifelong teacher to tell me how to be a good and a strong person.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Text classification is a fundamental text mining task with numerous real-life applications. While deep neural nets have achieved superior performance for text classification, they rely on large-scale labeled data to achieve strong performance. Obtaining large-scale labeled data, however, can be prohibitively expensive in many applications. In this project, we study the problem of learning neural text classifiers without using any labeled data, but only easy-to-provide heuristic rules as weak supervision. This problem is challenging because rule-induced weak labels are often noisy and incomplete. To address these challenges, we propose a model that can be learned from multiple weak supervision sources with two key components. The first component is a rule denoiser, which estimates conditional source reliability using a soft attention mechanism and reduces label noise by aggregating rule-induced noisy data. The second is a neural classifier that predicts soft labels for unmatchable samples to address the rule coverage issue. The two components are integrated into a co-training framework, which can be trained end-to-end to mutually enhance each other. We evaluate our model on five benchmarks for four popular text classification tasks, including sentiment analysis, topic classification, spam classification, and relation extraction. The results show that our model outperforms state-of-the-art weakly-supervised and semi-supervised methods, and achieves comparable performance with fully-supervised methods even without any labeled data.

CHAPTER 1
INTRODUCTION

Text is one of the basic and important data for human to use everyday, where 80% of human knowledge is stored in the text [1]. To better understand and manipulate human language, a field of artificial intelligence called Natural language processing (NLP) becomes very popular, which studies the use of computers to process human language for the purpose of performing tasks in a smart way [2]. Many NLP tasks can be formulated as text classification problems, such as sentiment analysis [3], topic classification [4], relation extraction [5] and question answering like slot filling [6]. Recent years have witnessed the rapid development of deep neural networks (DNNs) for this problem, from convolutional neural network (CNN, [7, 8]), recurrent neural network (RNN, [9]) to extra-large pre-trained language models [10, 11, 12]. DNNs' power comes from their capabilities of fitting complex functions based on large-scale training data.

In many applications, however, large-scale labeled data are unavailable and manually annotating data at a large scale can be prohibitively expensive. Labeling a large number of documents is a time consuming process requires a huge load of human labor. For example, in the Google Cloud labeling service, labeling one document requires about \$1 [1]. For many NLP tasks, the dataset is huge and if we label millions of documents, the cost will be extremely high. In such cases, the lack of training data has become the key bottleneck of applying DNNs for text classification.

Weakly-supervised learning is an attractive approach to address the data sparsity problem. It labels massive data with cheap labeling sources such as heuristic rules or knowledge bases, and leverages them to train the classifier. However, the major challenges of using weak supervision for text classification are two-fold: 1) the created labels are highly noisy

[1] https://cloud.google.com/ai-platform/data-labeling/pricing

and imprecise. The *label noise* issue arises because heuristic rules are often too simple to capture rich contexts and complex semantics for texts. For instance, while a rule 'Chicago $\rightarrow$ City' for the entity typing task is intuitive and often correct, sometimes it also lead to incorrect labels because the term 'Chicago' can also represent a Broadway show; 2) each source only covers a small portion of the data, leaving the labels incomplete. Seed rules have *limited coverage* because they are defined over the most frequent keywords but real-life text corpora often have long-tail distributions, so the instances containing only long-tail keywords cannot be annotated.

There have been studies [13, 14, 15, 16] that attempt to use weak supervision for deep text classification. Unfortunately, their performance is limited by the above two challenges. Snorkel is a famous platform of exploring weak supervision[13], which proposes a data programming method that uses labeling functions to automatically label data and then trains discriminative models with these labels. However, data annotated in this way only cover instances directly matched by the rules, making the model have limited performance on unmatched data.

We study the problem of using multiple weak supervision sources (e.g., domain experts, pattern matching) to address the challenges in weakly-supervised text classification. While each source is weak, multiple sources can provide complementary information for each other. There is thus potential to leverage these multiple sources to infer the correct labels by estimating source reliability in different feature regimes and then aggregating weak labels. Moreover, since each source covers different instances, it is more promising to leverage multiple sources to bootstrap on unlabeled data and address the label coverage issue.

Motivated by the above, we propose a model with two reciprocal components. The first is a *label denoiser* with the conditional soft attention mechanism [17] (section 4.2). Conditioned on input text features and weak labels, it first learns reliability scores for labeling sources, emphasizing the annotators whose opinions are informative for the particular corpus. It then denoises rule-based labels with these scores. The second component is a *neural*

classifier that learns labels and distributed feature representations for all samples, matched and unmatched. This neural classifier is supervised by both the denoised labels and its own confident predictions on the unmatched data, enabling it to solve the rule coverage problem while simultaneously enhancing the rule denoiser via patterns present in the unmatched data. These two components are integrated into an end-to-end co-training framework, benefiting each other through cross-supervision losses, including the rule denoiser loss, the neural classifier loss, and the self-training loss(section 4.4).

We evaluate our model on four classification tasks, including sentiment analysis, topic classification, spam classification, and information extraction. The results on five benchmarks show that: 1) the soft-attention module effectively denoises the noisy training data induced from weak supervision sources, achieving 84% accuracy for denoising; and 2) the co-training design improves prediction accuracy for unmatched samples, achieving at least 9% accuracy increase on them. In terms of the overall performance, our model consistently outperforms advanced weakly supervised methods [13, 14, 15], semi-supervised method [18], and fine-tuning method [19] by 5.46% on average.

Our contributions are summarized as follows:

- We propose a neural co-training framework that leverages multi-source weak supervision for deep text classification. The co-training design jointly models multiple weak supervision sources to learn a text classifier in an end-to-end manner. Such a co-training framework makes it possible to learn accurate deep text classifiers without any labeled data.

- We propose a label denoising module that can reduce label noise in weak supervision. Based on a conditional soft attention mechanism to encode both data information and source knowledge, the denoising module estimates source reliability to remove label noise effectively. In addition, we propose a self-trainable neural classifier, which tackles the label coverage issue b bootstrapping on unlabeled data.

- We perform extensive experiments on five different benchmarks for four text classification tasks. The results demonstrate our model outperforms strong baselines, and achieves comparable performance to fully supervised models without using any labeled data.

In the following parts, Chapter 2 to 5 describe our proposed model related work for learning from multi-source weak supervision for neural text classification. Chapter 2 Introduces related work and related techniques. Chapter 3 states the preliminaries of our studying problem. Chapter 4 explains our model with both a overview and detailed interpretation of each module. Chapter 5 shows the experiments and discusses the results. Finally, Chapter 6 sums up our work and briefly indicate several future work directions.

CHAPTER 2

BACKGROUND

2.1 Related Work

2.1.1 Text classification Overview

The text classification task is defined as automatically assign labels to a document [20]. We begin with a brief overview of classic text classification methods. Generally, text classification systems consists of four steps: feature extraction, dimension reductions, classification, and evaluations [20]. At the first step, a document is expressed as a high dimension vector, and common techniques of text representation include Bag-of-words (BoW) [21], which represent a document as a histogram of word occurrences; Term Frequency-Inverse Document Frequency (TF-IDF) [22], a term-weighting scheme determines how relevant a given word is in one document. Then, dimension reductions methods like Principal Component Analysis (PCA) [23] and Linear Discriminant Analysis (LDA) [24] are applied for preparing the classification. Popular text classifiers consists of k-nearest neighbor (KNN) [25], Support Vector Machine (SVM) [26] , Random Forest [27], and Naive Bayes Classifier (NBC) [28]. Finally, metrics like Accuracy [29], Receiver Operating Characteristics (ROC) [30], and Matthews Correlation Coefficient (MCC) [31] are used to evaluate how a model performs.

Deep Neural Networks (DNNs) have recently been adapted to natural language processing (NLP) tasks. In the feature representation phase, Word2Vec [32] is a milestone work to produce word embeddings. This approach uses shallow, two-layer neural networks with the continuous bag-of-words (CBOW) and the Skip-gram model to train a high dimension vector for each word. Other powerful word embedding methods includes GloVe [33], FastText [34] and so on. Moreover, DNNs also can combine the feature engineering and

classification steps. Convolution neural networks (CNNs) make use of the "convolutional filters" to automatically extract features corresponding to specific tasks. They are currently used for sentiment analysis and opinion mining ([4]) for short texts with fairly balanced class distributions. Recurrent Neural Networks (RNNs) using the Long Short Term Memory (LSTM) architecture are ideal for text and speech analysis, which take advantage of sequential information [35]. Hierarchical Attention Network (HAN) [36] is a deep architecture with two levels of attention mechanisms applied at both the word and the sentence level, attempting to attend the importance of context for better classification.

With the growth of deep learning, the number of model parameters has increased rapidly due to the deeper and deeper neural network architectures, which requires a much larger dataset to fully fit model parameters [37]. However, as mentioned in Chapter 1, labeling large-scale datasets is a big issue because of the huge annotation costs. To address the challenge, leveraging the huge unlabeled text data to build pre-trained models and learning from multiple weak sources are two promising ways.

2.1.2 Pre-trained Models (PTMs)

Pre-training on the large-scale text corpus is an effective approach to learn universal language representations, which are then beneficial for downstream NLP tasks to avoid huge parameter training from scratch by fine-tuning only relatively shallow layers (*e.g.*, 1 - 3 neural layers) for the specific tasks [37]. The above-mentioned word embedding models like Word2Vec can be viewed as the early step of PTMs. Recently, PTMs focus more on the contextual level to learn the better representation of text.

Peters *et al.*[38] proposed ELMo (Embeddings from Language Models) stands for the representations output by a pre-trained 2-layer LSTM encoder with a bidirectional (*i.e.*forward and backward) language model (BiLM). ELMo achieved a big improvement on many NLP benchmarks due to the powerful representation ability. However, such models are usually applied as feature extractors so their parameters are fixed when feeding into

downstream tasks. The rest parts of the main model still require training from scratch.

To address this issue, ULMFiT (Universal Language Model Fine-tuning) [19] proposed to fine-tune pre-trained language models (LMs) and achieved state-of-the-art results on text classification tasks. This model first pre-trains LMs on the large scale general-domain corpus and then fine-tunes LMs on the target data and target task. Since then, fine-tuning has been the mainstream choice to adjust PTMs for the downstream tasks.

Nowadays, with the growth of computation abilities, the extreme deep PTMs like OpenAI GPT (Generative Pre-training) [39] and BERT (Bidirectional Encoder Representation from Transformer) [10] are very popular in many NLP tasks with their powerful ability in getting universal language representations.

2.1.3 Learning from Multi-Source Supervision

The crowdsourcing area also faces the problem of learning from multiple sources (*i.e.*, crowd workers). Different strategies have been proposed to integrate the annotations for the same instance, such as estimating the confidence intervals for workers [40] or leveraging approval voting [41]. Compared with crowdsourcing, our problem is different in that the multiple sources provide only feature-level noisy supervision instead of instance-level supervision.

More related to our work are data programming methods [42, 13, 43] that learn from multiple weak supervision sources. One seminal work in this line is Snorkel [13], which treats true labels as latent variables in a generative model and weak labels as noisy observations. The generative model is learned to estimate the latent variables, and then the denoised training data are used to learn classifiers. Our approach differs from data programming methods in two ways. First, instead of using generative models to estimate the latent clean labels, we use a soft attention mechanism to estimate source reliability, which can be integrated into neural text classifiers and trained end-to-end. Second, data programming methods may suffer from limited performance on unmatched samples, while our approach

addresses this issue by jointly training a label denoiser and neural classifier.

Other advanced related work attempts to combine the multiple sources in different ways to make use of comprehensive information and denoise. Meng *et al.*[14] proposes a deep self-training method that uses weak supervision to learn an initial model and updates the model by its own confident predictions. However, the self-training procedure can overfit the label noise and is prone to error propagation. Zamani *et al.*[15] solves query performance prediction (QPP) by boosting multiple weak supervision signals in an unsupervised way. However, they choose the most informative labelers by an ad-hoc user-defined criterion, which may not generalize to all the domains. Awasthi *et al.*[16] assumes that human labelers are over-generalized to increase the coverage, and they learn restrictions on the rules to address learning wrongly generalized labels. However, their method requires the specific formulation process of rules to indicate which rules are generated by which samples, so that it cannot deal with other kinds of labeling sources like knowledge bases or third-party tools.

2.1.4 Learning from Noisy Supervision

Our work is closely related to existing work on learning from noisy supervision. To deal with label noise, several studies [44, 45, 46] adopt a data cleaning approach that detects and removes mislabeled instances. This is achieved by outlier detection [44], prior heuristics [45], or reinforcement learning [46]. One drawback of this data cleaning approach is that it can discard many samples and incur information loss.

Different from data cleaning, some works adopt a data correction approach. The most prominent idea in this line is to estimate the noise transition matrix among labels [47, 48, 49], and then use the transition matrices to re-label the instances or adapt the loss functions. Meanwhile, re-weighting strategies have also been explored to adjust the input training data. These techniques weigh training samples according to the confidence of predictions [50], one-sided noise assumption [51], or the similarity of their descent directions [46].

Recently, a few studies [52, 53] have also explored designing denoising modules for neural networks. However, our method differs from them in that: (1) our method learns *conditional* reliability scores for multiple sources; and (2) these methods still require clean data for denoising, while ours does not.

2.1.5 Self-training and Co-training

Self-training is a classic technique for learning from limited supervision [54]. The key idea is to use a model's confident predictions to update the model itself iteratively. However, one major drawback of self-training is that it is sensitive to noise, *i.e.*, the model can be misguided by its own wrong predictions and suffer from error propagation. This issue is particularly common for deep neural networks because they can easily overfit label noise and be over-confident in their predictions [55].

Co-training [56] extends self-training by allowing two classifiers to exchange their expertise until they reach a consensus. This is typically be achieved by letting the two classifiers annotate unlabeled samples with their confident predictions and update each other. Recently, the co-training idea has been extended to deep neural networks. Qiao *et al.*[57] used co-training to train two networks that have consistent predictions over all the samples, and meanwhile make each network resistant to the adversarial examples from its peer network to prevent them from collapsing into each other. Laine *et al.*[58] used different regularizations and input augmentation conditions to improve deep co-training. Chen *et al.*[59] proposed the Tri-Net model, which uses output smearing to initialize modules and then fine-tunes on labeled data to augment model diversity. Compared with these self-training and co-training methods, our method alleviates the error propagation issue by estimating conditional source reliability and performing weighted majority voting. Moreover, these methods are designed for semi-supervised settings with clean supervision, whereas our method learns from noisy supervision.

2.2 Related Techniques

2.2.1 Attention Mechanisms

sequence-to-sequence (Seq2Seq) models, consisting of an encoder and a decoder, has been successful applied to many NLP tasks [35]. However, traditional Seq2Seq methods view each words share the same importance in a input sentence. However, in a document, not all the tokens play equal roles for answering a question or determining the sentiment. To solve this problem, attention mechanism attempts to extract specific words that have the high importance to the meaning of the sentence, and then aggregate the representation of these informative words to generate a sentence embedding. Specifically,

$$\begin{aligned} u_{it} &= \tanh\left(W_w h_{it} + b_w\right) \\ \alpha_{it} &= \frac{\exp\left(u_{it}^\top u_w\right)}{\sum_t \exp\left(u_{it}^\top u_w\right)} \\ s_i &= \sum_t \alpha_{it} h_{it} \end{aligned} \tag{2.1}$$

First, the word passes one neural layer to git its hidden representation u_{it}. Then, the importance score of this word is measure together with a word-level context vector u_w. The softmax function helps to get the normalized weight α_{it}. Finally, the weighted sum of the word representations form the sentence vector s_i as the embedding of a sentence [36].

2.2.2 BERT

BERT stands for Bidirectional Encoder Representations from Transformers. It is designed to pre-train deep bidirectional representations from a large-scale unlabeled corpus. To achieve powerful representations, it is pre-trained on two difficult NLP tasks: Masked Language Modeling and Next Sentence Prediction. As a result, without making any major change but only fine-tuned with one additional output layer, the performance is very promising on multiple kinds of NLP tasks [10]. In this project, we directly use the pre-

trained BERT to get the embeddings of documents without fine-tuning, which is very time and computing efficiency.

CHAPTER 3

PRELIMINARIES

3.1 Problem Definition

Let $\mathcal{D} = \{\mathbf{d}_1, \ldots, \mathbf{d}_n\}$ be a corpus of text documents, and $\mathcal{C} = \{\mathcal{C}_1, \ldots, \mathcal{C}_m\}$ be a set of target classes. Here, the term 'documents' is a general notion and can refer to various forms of text elements. For example, in topic classification, a document is an entire document and the target labels are topic categories; in entity typing, a document is an entity mention along with its context and the target labels are entity types; in relation classification, a document is a sentence containing two entities, and the target labels are relation types. Given the text corpus $\mathcal{D}$ and the label set $\mathcal{C}$, text classification aims to assign a class label $\mathcal{C}_j \in \mathcal{C}$ for each document $\mathbf{d}_i \in \mathcal{D}$.

Existing supervised text classification methods assume that a set of well-annotated training data are available and use them to train a text classifier. However, in many applications, obtaining large-scale well-annotated data is expensive. To break this bottleneck, we study the *weakly supervised* text classification problem. In this setting, an external source can provide a set of labeling rules as weak supervision, defined as follows:

Definition 1 (Weak Supervision)**.** A weak supervision source is an oracle that specifies a set of labeling rules $\mathcal{R} = \{r_1, r_2, \ldots, r_k\}$. Each rule r_i declares a mapping $f \rightarrow \mathcal{C}$, meaning that any documents that contain feature f must belong to class $\mathcal{C}$.

Weak supervision can automatically create training data and largely reduce labeling efforts for text classification. However, using one weak supervision source often leads to limited performance. We thus assume there are multiple weak supervision sources providing complementary information to each other. A concrete example is provided below.

Example 1 (Multi-Source Weak Supervision)**.** Consider a sentiment analysis problem on

a corpus of Yelp reviews. Figure 3.1 shows three weak supervision sources, where the sources use labeling rules to encode domain knowledge from different aspects. For example, the second source provides rules from the 'service' aspect, declaring rules such as 'rude $\rightarrow$ negative' and 'friendly $\rightarrow$ positive'. For each source, its rules translate into a set of 'if-else' labeling functions, such that any samples matching any predicates are automatically annotated with the corresponding labels. Meanwhile, the samples that cannot match any rules will remain unlabeled.

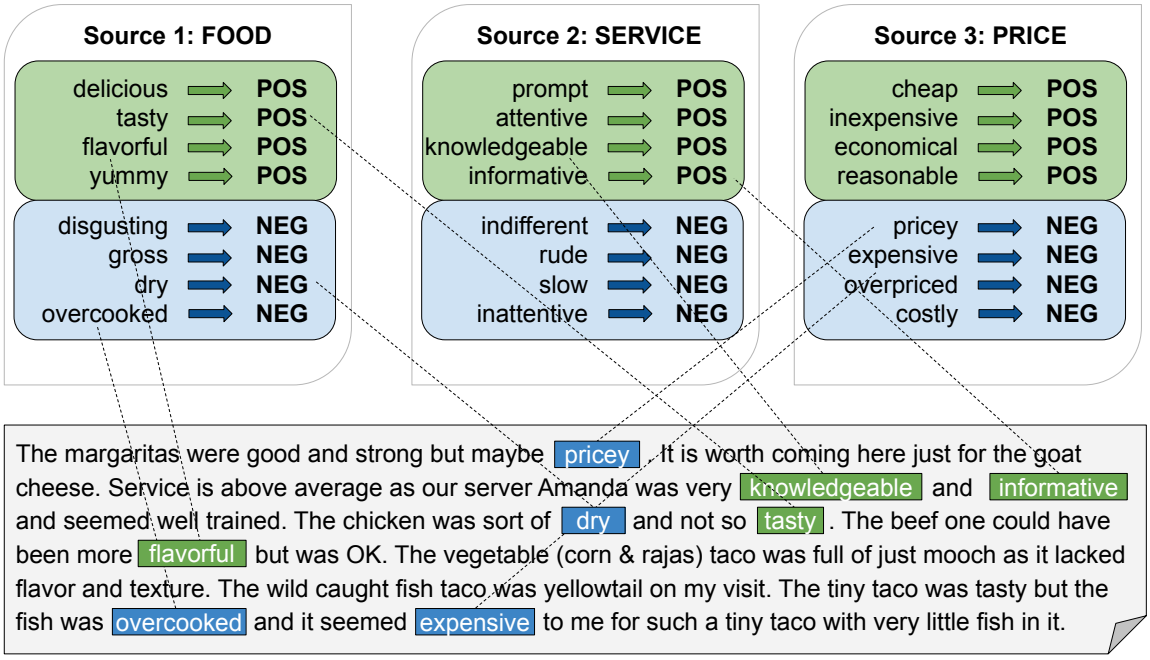


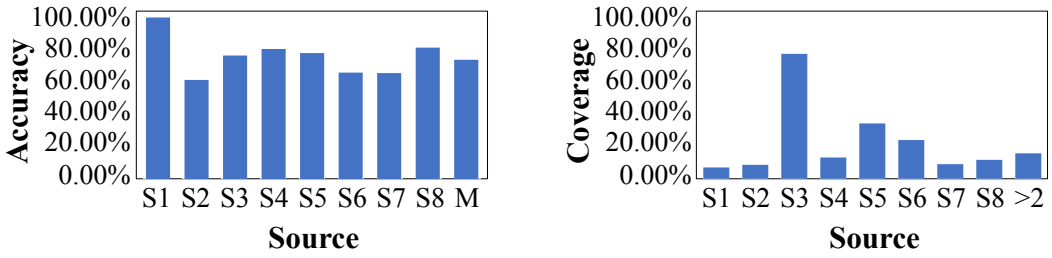
Figure 3.1: The annotation process for three weakly supervision sources. "POS" and "NEG" are the labels for the sentiment analysis task.

Problem Formulation. We now formulate the multi-source weakly supervised text classification problem. At a high level, this problem aims to leverage only multiple weak supervision sources and unlabeled data to achieve accurate text classifications. Formally, we have: (1) a corpus $\mathcal{D}$ of text documents; (2) a set $\mathcal{C}$ of target classes; and (3) a set $\mathcal{S} = \{\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_k\}$ of weak annotators. This problem has two settings: *inductive classification* and *transductive classification*. In the inductive setting, our goal is to learn a classifier from $\mathcal{D}$ to accurately classify any newly arriving documents. In the transductive

setting, our goal is to learn a classifier from $\mathcal{D}$ to accurately classify the documents in $\mathcal{D}$ into the classes in $\mathcal{C}$.

3.2 Challenges

The annotations automatically created from different weak labelers can largely reduce human labeling efforts. However, rule-induced labeled data has two drawbacks: *label noise* and *label incompleteness*.



(a) the accuracy of the rules

(b) the coverage of the rules

Figure 3.2: The accuracy and coverage of rule-induced data on the Yelp dataset. We specify eight intuitive rules for the dataset and compute.

Challenge 1: Label Noise. Rule-induced labeled data are noisy since user-provided rules are often simple and don't fully capture complex patterns and semantics of human language. We performed a sentiment analysis case study on a Yelp Review dataset using eight sources of weak supervision from both intuitive and machine learning-based rule. The details of the rules can be found in the supplementary material. However, rigorous investigation shows that these rules have rather poor accuracy (68.3% on average), which is only marginally increased to 71.6% by majority voting. Figure 3.2(a) further illustrates that rules with high accuracy suffer most from poor coverage, thus preventing them from labeling a meaningful proportion of samples.

Label noise hurts the performance of text classifiers - especially deep classifiers - because such complex models easily overfit the noise. A ULMFiT sentiment model trained on Yelp's noisy labels obtained a meager test accuracy of 67.3%, far below the 89.6% obtained

with clean labels. To effectively leverage multi-source weak supervision, it is challenging yet important to reduce label noise.

Challenge 2: Label Incompleteness. The second challenge is that the rules have limited coverage. User-provided labeling rules are specified over common lexical features, but real-life data are long-tailed. Continuing the Yelp case study, we see in Figure 3.2(b) that coverage ranges from 6.8% to 22.2%, with the exception of one rule that uses very general keywords (*e.g.*'great','bad'). Worse, only 15.3% of samples are covered by at least two rules.

Low coverage particularly limits the performance of text classifiers on unmatched samples. For the Yelp sentiment analysis task, we aggregated the weakly annotated data and trained a BERT-based [10] neural classifier which achieved 89.41% accuracy on samples matched by at least three rules, but only 79.54% on samples matched one rule, and 76.54% unmatched samples.

CHAPTER 4

METHODOLOGY

In this section, we describe our proposed method for learning neural classifiers from multiple weak supervision sources. We begin with an overview of our method in section 4.1 and then introduce its two key components in section 4.2 and section 4.3. We describe the co-training objective in section 4.4 and present the model learning and inference procedures in section 4.5.

4.1 The Overall Framework

Our method addresses the above challenges by integrating weak annotated labels from multiple sources and text data to an end-to-end framework with a label denoiser and a deep neural classifier, illustrated in Figure 4.1.

Label denoiser & self-denoising We handle the label noise issue by building a label denoiser that iteratively denoises itself to improve the quality of weak labels. This label denoiser estimates the source reliability using a conditional soft attention mechanism, and then aggregates weak labels via weighted voting of the labeling sources to achieve "pseudo-clean" labels. The reliability scores are conditioned on both rules and document feature representations. They effectively emphasize the opinions of informative sources

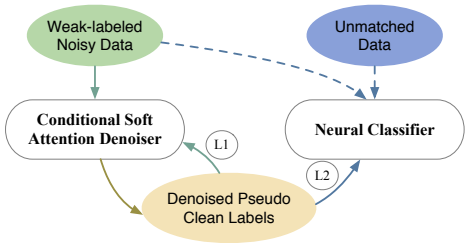


Figure 4.1: Overview of cross-training between the rule-based classifier and the neural classifier.

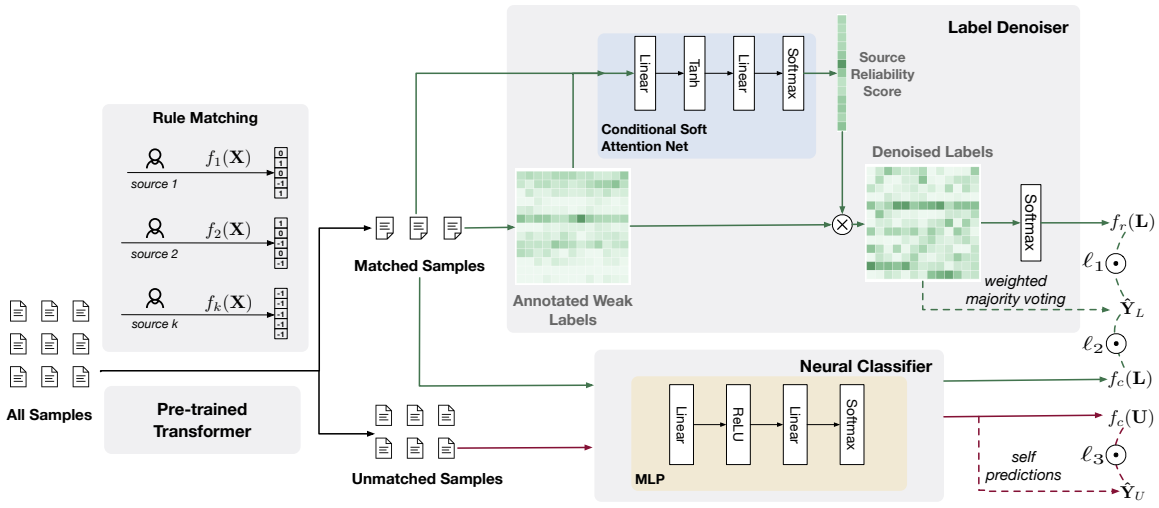


Figure 4.2: The detailed model architecture. Our model mainly consists of two parts: (1) the label denoiser, including the conditional soft attention reliability estimator and the instance-wise multiplication; (2) the neural classifier, which calculates sentence embedding using the pre-trained Transformer and makes classification.

while down-weighting those of unreliable sources, thus making rule-induced predictions more accurate.

Neural classifier & self-training To address the low coverage issue, we build a neural classifier which learns distributed representations for text documents and classifies each of them, whether rule-matched or not. It is supervised by both the denoised weakly labeled data as well as its own high-confident predictions of unmatched data.

The training objective. The label denoiser and the neural classifier are integrated into an end-to-end training framework to iteratively enhance each other. The details for each loss is introduced later in section 4.4.

4.2 The Label Denoiser

When aggregating multiple weak supervision sources, it is key for the model to attend to more reliable sources, where source reliability should be conditioned on input features. This will enable the model to aggregate multi-source weak labels more effectively. Given

k labeling resources, we obtain the weak label matrix $\tilde{Y} \in \mathbb{R}^{n \times k}$ through rule matching. We then estimate the source reliability and aggregate complementary weak labels to obtain "pseudo-clean" labels.

Parameterize the source reliability We introduce a soft attention mechanism conditioned on both weak labels and feature representation, denoted as $\boldsymbol{B}$, to estimate the source reliability. Formally, we denote the denoised "pseudo-clean" labels by $\hat{Y} = [\hat{y_1}, \ldots, \hat{y_n}]^T$, and the initial ones $\hat{Y}_0$ are obtained by simple majority voting from $\tilde{Y}$.

The core of the label denoiser is an attention net, a two-layer feed-forward neural network which predicts the attention score for matched samples. Formally, we specify a reliability score a_j for each labeling source to represent its annotation quality, and the score is normalized to satisfy $\sum_{j=1}^{k} a_j = 1$. For one document $\boldsymbol{d_i}$, its attention score $q_{i,j}$ of one labeling source $\mathcal{R}_j$ is:

$$\begin{aligned} \hat{q}_{ij} =& W_2^T \tanh(W_1(\tilde{y}_{ij} + \boldsymbol{B}_i)), \\ q_{ij} =& \frac{\exp(\hat{q}_{ij})}{\sum_j \exp(\hat{q}_{ij})}, \end{aligned} \tag{4.1}$$

where W_1, W_2 denote the neural network weights and $\tanh$ is the activation function. Thus, for each document, its conditional labeling source score vector $\boldsymbol{A}_i = [a_{i1}, a_{i2}, \ldots, a_{ik}]^T$ is calculated over matched annotators as $a_{ij} = q_{ij}\chi_C(\tilde{y}_{ij} >= 0)$, where χ_C is the indicator function. Then, we average the conditional source score $\boldsymbol{A}_i$ over all the n matched samples to get the source reliability vector $\boldsymbol{A}$. The weight of j_{th} $(j = 1, 2, \ldots, k)$ annotator is calculated as $a_j = \frac{1}{n}\sum_{i=1}^{n} a_{ij}$. Finally, We aggregate k reliability scores to get the reliability vector $\boldsymbol{A} = [a_1, a_2, \ldots, a_k]^T$.

Denoise pseudo labels With the learned reliability vector $\boldsymbol{A}$, we reweight the sources to get the weighted majority voted labels $\hat{Y}$ by $\tilde{Y_i} \otimes \boldsymbol{A}$. The denoised "pseudo-clean" label $\hat{y_i}$

is:

$$\hat{y}_i = \arg\max_{C_r} \sum_{j=1}^{k} a_j \chi_C(\tilde{y}_{ij} == C_r), \quad \text{where } r = 1, 2, \ldots, m. \tag{4.2}$$

The updated higher-quality labels $\hat{Y}$ then supervise rule-covered samples $\mathcal{D}_L$ to generate better soft predictions and guide the neural classifier later.

Rule-based classifier prediction At the epoch t of our co-training framework, we learn the reliability score $\boldsymbol{A}(t)$ and soft predictions $\hat{\boldsymbol{Z}}(t)$ supervised by "pseudo-clean" labels from the previous epoch $\hat{Y}(t-1)$. Then we renew "pseudo-clean" labels as $\hat{Y}(t)$ using the score $\boldsymbol{A}(t)$ by (Equation 4.2).

Specifically, given m target classes and k weak annotators, the prediction probability $\hat{\boldsymbol{z}}_{\boldsymbol{i}}$ for $\boldsymbol{d}_{\boldsymbol{i}}$ is obtained by weighting the noisy labels $\tilde{Y}_i$ according to their corresponding conditional reliability scores $\boldsymbol{A}_i$: $\hat{\boldsymbol{z}}_{\boldsymbol{i}} = \text{softmax}(\tilde{Y}_i \otimes \boldsymbol{A}_i)$, where the masked matrix multiplication $\otimes$ (defined in (Equation 4.3)) is used to mask labeling sources that do not annotate document i, and we normalize the resultant masked scores via softmax:

$$\begin{aligned} y_{ir} &= \sum_{j=1}^{k} a_{ij} \chi_C(\tilde{y}_{ij} == C_r) \\ \hat{\mathbf{z}}_{ir} &= \frac{\exp(y_{ir})}{\sum_{r=1}^{m} \exp(y_{ir})}. \end{aligned} \tag{4.3}$$

We finally aggregate m soft adjusted scores to get the soft prediction vector $\hat{\boldsymbol{z}}_{\boldsymbol{i}} = [z_{i1}, \ldots, z_{im}]^T$.

4.3 The Neural Classifier

The neural classifier is designed to handle all the samples, including matched ones and unmatched ones. The unmatched corpus where the documents cannot be annotated by any source is denoted as $\mathcal{D}_U$. In our model, we use the pre-trained BERT [10] as our feature extractor, and then feed the text embeddings into a feed-forward neural network to obtain the final predictions. For $\boldsymbol{d}_i \in \mathcal{D}_L \cup \mathcal{D}_U$, the prediction $\tilde{\boldsymbol{z}}_{\boldsymbol{i}}$ is:

$$\tilde{\boldsymbol{z}}_{\boldsymbol{i}} = f_\theta(\boldsymbol{B}_i; \theta_w), \tag{4.4}$$

where f_θ denotes the two-layer feed-forward neural network, and θ_w denotes its parameters.

4.4 The Training Objective

The rule denoiser loss ℓ_1 is the loss of the rule-based classifier over $\mathcal{D}_L$. We use the "pseudo-clean" labels $\hat{Y}$ to self-train the label denoiser and define the loss ℓ_1 as the negative log likelihood of $\hat{y}_i$:

$$\ell_1 = -\sum_{i \in \mathcal{D}_L} \hat{y}_i \log \hat{\boldsymbol{z}}_{\boldsymbol{i}}. \tag{4.5}$$

The neural classifier loss ℓ_2 is the loss of the neural classifier over $\mathcal{D}_L$. Similarly, we regard the negative log-likelihood from the neural network outputs $\tilde{\boldsymbol{Z}}$ to the pseudo-clean labels $\hat{Y}$ as training loss, namely

$$\ell_2 = -\sum_{i \in \mathcal{D}_L} \hat{y}_i \log \tilde{\boldsymbol{z}}_{\boldsymbol{i}}. \tag{4.6}$$

The unsupervised self-training loss ℓ_3 is the loss of the neural classifier over $\mathcal{D}_U$. To further enhance the label quality of $\mathcal{D}_U$ we apply the temporal ensembling strategy [58], which aggregates the predictions of multiple previous network evaluations into an ensemble prediction to alleviate noise propagation. For a document $\boldsymbol{d}_i \in \mathcal{D}_U$, the neural classifier outputs $\tilde{\boldsymbol{z}}_{\boldsymbol{i}}$ are accumulated into ensemble outputs $\boldsymbol{Z}_{\boldsymbol{i}}$ by updating $\boldsymbol{Z}_{\boldsymbol{i}} \leftarrow \alpha \boldsymbol{Z}_{\boldsymbol{i}} + (1-\alpha)\tilde{\boldsymbol{z}}_{\boldsymbol{i}}$, where α is a term that controls how far the ensemble looks back into training history. We also need to construct target vectors by bias correction, namely $\boldsymbol{p}_{\boldsymbol{i}} \leftarrow \boldsymbol{Z}_{\boldsymbol{i}}/(1-\alpha^t)$, where t is the current epoch. Then, we minimize the Euclidean distance between $\boldsymbol{p}_{\boldsymbol{i}}$ and $\tilde{\boldsymbol{z}}_{\boldsymbol{i}}$, where

$$\ell_3 = \sum_{i \in \mathcal{D}_U} \quad \|\tilde{\boldsymbol{z}}_{\boldsymbol{i}} - \boldsymbol{p}_i\|^2. \tag{4.7}$$

Overall Objective The final training objective is to minimize the overall loss ℓ:

$$\ell = c_1\ell_1 + c_2\ell_2 + c_3\ell_3, \tag{4.8}$$

Algorithm 1 Training process of our model

Require: $\mathcal{D}_L, \mathcal{D}_U, \mathcal{C}, \boldsymbol{B}, \tilde{Y}, g_w(x)$ and $f_\theta(x)$: feed-forward rule-based and nerual classifier with trainable parameters W and θ; s: number of training iteraions;

1: $\hat{Y} \leftarrow \hat{Y}_0$, initialize by simple majority voting
2: **for** $t \leftarrow 1$ to s **do**
3: $\quad \boldsymbol{A}, \hat{\boldsymbol{z}}_{i \in \mathcal{D}_L} \leftarrow g_w(\tilde{Y}_i, \boldsymbol{B}_i, \hat{y}_i)$ ▷ learn reliability score and evaluate attention network output supervised by "pseudo-clean" labels from (Equation 4.1) and (Equation 4.3)
4: $\quad \hat{y}_i \leftarrow$(Equation 4.2) ▷ renewed pseudo labels
5: $\quad \tilde{\boldsymbol{z}}_{i \in \mathcal{D}_L \cup \mathcal{D}_L} \leftarrow f_\theta(\boldsymbol{B_i}, \hat{y}_i)$ ▷ evaluate neural classifier output
6: $\quad$ update θ, W using ADAM by (Equation 4.8)
7: **end for**
8: **return** W, θ

where $0 \leq c_1 \leq 1$, $0 \leq c_2 \leq 1$, and $0 \leq c_3 \leq 1$ are hyper-parameters for balancing the three losses and satisfy $c_1 + c_2 + c_3 = 1$.

4.5 Model Learning and Inference

Algorithm 1 summarizes the overview model learning steps, and Algorithm 2 sketches the detailed training procedure. We can see that the two classifiers provide supervision signals for both themselves and their peers, iteratively improving their classification abilities.

In the test phase, the test corpus is sent into our model with the corresponding annotated noisy labels. The final target $\mathcal{C}_i$ for a document i is predicted by ensembling the soft predictions. If two predictions from the label denoiser and the neural classifier conflict with each other, we choose the one with higher confidence, where the confidence scores are softmax outputs.

Algorithm 2 Training process of our model

Require: $\mathcal{D}_L$: a set of matched training documents indices with known labels (major voting labels); $\mathcal{D}_L$: a set of unmatched training documents indices without known labels; C: a set of m target classes;

Require: $\boldsymbol{B}$: the representations obtained by the pre-trained Transformer;

Require: $\tilde{Y}$: set of annotated labels by weakly supervision sources

Require: c_1, c_2, c_3: weight scales;

Require: $g_w(x)$: feed-forward rule-based classifier with trainable parameters W; $f_\theta(x)$: feed-forward neural classifier with trainable parameters θ;

Require: s: total training epoch numbers;

Require: $\hat{Y}_0$: "pseduo-clean" majority voted labels from sources

$\hat{Y} \leftarrow \hat{Y}_0$ ▷ initialize "pseudo-clean" labels for $\mathcal{D}_L$

$\boldsymbol{Z} \leftarrow \mathbf{0}_{[n \times m]}$ ▷ initialize ensemble predictions for self-training

$\boldsymbol{p} \leftarrow \mathbf{0}_{[n \times m]}$ ▷ initialize target vectors for self-training

for $t \leftarrow 1$ to s **do**

  $\boldsymbol{A}, \hat{\boldsymbol{z}}_{i \in \mathcal{D}_L} \leftarrow g_w(\tilde{Y}_i, \boldsymbol{B}_i, \hat{y}_i)$ ▷ learn reliability score conditioned on annotators and document representations; evaluate attention network output supervised by "pseudo-clean" labels

  $\hat{y}_i \leftarrow \arg\max_{C_m} \sum_{j=1}^{k} a_j \chi_C \left(\tilde{y}_{ij} = C_m\right)$ ▷ renewed pseudo labels with score $\boldsymbol{A}$

  $\tilde{\boldsymbol{z}}_{i \in \mathcal{D}_L \cup \mathcal{D}_L} \leftarrow f_\theta(\boldsymbol{B_i}, \hat{y}_i)$ ▷ evaluate self-learning network output with renewed "pseudo-clean" labels

  $\ell_1 \leftarrow -\sum_{i \in \mathcal{D}_L} \hat{y}_i \log \hat{\boldsymbol{z}}_{\boldsymbol{i}}$ ▷ supervised loss component for rule-based classifier

  $\ell_2 \leftarrow -\sum_{i \in \mathcal{D}_L} \hat{y}_i \log \tilde{\boldsymbol{z}}_{\boldsymbol{i}}$ ▷ supervised loss component for neural classifier

  $\ell_3 \leftarrow \sum_{i \in \mathcal{D}_U} \|\tilde{\boldsymbol{z}}_{\boldsymbol{i}} - \boldsymbol{p}\|^2$ ▷ self-training unsupervised loss

  $Loss \leftarrow (1 - c_2 - c_3) L_1 + c_2 L_2 + c_3 L_3$ ▷ total loss

  $\boldsymbol{Z} \leftarrow \alpha \boldsymbol{Z} + (1 - \alpha) \tilde{\boldsymbol{z}}$ ▷ accumulate ensemble predictions

  $\boldsymbol{p} \leftarrow Z / (1 - \alpha^t)$ ▷ construct target vectors

  update θ, W using ADAM ▷ update network parameters

end for

return W, θ

CHAPTER 5
EXPERIMENTS AND RESULTS

In this section, we evaluate the empirical performance of our model. We conduct experiments to answer the following questions:

1. how effective is our model compared to state-of-the-art weakly supervised text classifiers and fully-supervised text classifiers?
2. how effective is our model in reducing label noise in the rule-induced data?
3. how is the performance of our model on the samples uncovered by rules?
4. how effective is our end-to-end training framework?

5.1 Experimental Setup

Datasets and tasks We evaluate our model on five widely-used text classification datasets, covering four different text classification tasks: **youtube** [60] (Spam Detection), **imdb** [61], **yelp** [4] (Sentiment Analysis), **agnews** [4] (Topic Classification), and **spouse** [13] (Relation Classification). Table 5.1 shows the statistics of these datasets and the quality of weak labels (the details of each annotation rule are given in the section A.2). Creating such rules required very light efforts, but is able to cover a considerable amount of data samples (*e.g.*, 54k in agnews).

Baselines We compare our model with the following advanced methods: 1) **Snorkel** [13] is a general weakly-supervised learning method that learns from multiple sources and denoise weak labels by a generative model; 2) **WeSTClass** [14] is a weakly-supervised text classification model based on self-training; 3) **ImplyLoss** [16] propose the rule-exemplar

Table 5.1: Data Statistics. C is the number of classes. Cover is fraction of rule-induced samples. Acc. refers to precision of labeling sources (number of correct samples / matched samples). Cover and Acc. are in %.

Dataset	Task	C	#Train	#Dev	#Test	Cover	Acc.
youtube	Spam	2	1k	0.1k	0.1k	74.4	85.3
imdb	Sentiment	2	20k	2.5k	2.5k	87.5	74.5
yelp	Sentiment	2	30.4k	3.8k	3.8k	82.8	71.5
agnews	Topic	4	96k	12k	12k	56.4	81.4
spouse	Relation	2	1k	0.1k	0.1k	85.9	46.5

supervision and implication loss to denoise rules and rule-induced labels jointly; 4) **NeuralQPP** [15] is a boosting prediction framework which selects useful labelers from multiple weak supervision signals; 5) **MT** [18] is a semi-supervised model that uses Mean-Teacher method to average model weights and add a consistency regularization on the student and teacher model; and 6) **ULMFiT** [19] is a strong deep text classifier based on pre-training and fine-tuning. 7) **BERT-MLP** takes the pre-trained Transformer as the feature extractor and stacks a multi-layer perceptron on its feature encoder.

5.2 Experimental Results

5.2.1 Comparison with Baselines

We first compare our method with the baselines on five datasets. For fair comparison, all the methods use a pre-trained BERT-based model for feature extraction, and use the same neural architecture as the text classification model. All the baselines use the same set of weak labels $\tilde{Y}$ for model training, except for WeSTClass which only requires seed keywords as weak supervision (we extract these keywords from the predicates of our rules).

Table 5.2 shows the performance of all the methods on five datasets. As shown, our model consistently outperforms all the baselines across all the datasets. Such results show the strength and robustness of our model. Our model is also very time-efficient (4.5 minutes on average) with trainable parameters only from two simple MLP neural networks (0.199M trainable parameters).

Table 5.2: Classification accuracy in the test set for all methods on five datasets.

Method	**youtube**	**imdb**	**yelp**	**agnews**	**spouse**
Snorkel	78.6	73.2	69.1	62.9	56.9
WeSTClass	65.1	74.7	76.9	82.8	56.6
Imployloss	93.6	51.1	76.3	68.5	68.3
NeuralQPP	85.2	53.6	57.3	69.5	74.0
MT	86.7	72.9	71.2	70.6	70.7
ULMFiT	56.1	70.5	67.3	66.8	72.4
BERT-MLP	77.0	72.5	81.5	75.8	70.7
Ours	**94.9**	**82.9**	**87.5**	**85.7**	**81.3**

Similar to our methods, Snorkel, NeuralQPP, and Imployloss also denoise the weak labels from multiple sources: 1) Snorkel uses a generative modeling approach; 2) Imployloss adds one regularization to estimate the rule over-generalizing issue, but it requires the clean data to indicate which document corresponds to which rule. Without such information in our setting, this advanced baseline cannot perform well; 3) NeuralQPP selects the most informative weak labelers by boosting. The performance gaps verify the effectiveness of the our conditional soft attention design and the co-training framework.

WeSTClass is similar to our method in that it also uses self-training to bootstrap on unlabeled samples to improve its performance. The major advantage of our model over WeSTClass is that it uses two different predictors (rule-based and neural classifier) to regularize each other. Such a design not only better reduces label noise but also makes the learned text classifier more robust.

Finally, ULMFiT and BERT-MLP are strong baselines based on language model fine-tuning. MT is a well-known semi-supervised model which achieved inspiring results for image classification. However, in the weakly supervised setting, they do not perform well due to label noise. The results show that ULMFiT and MT suffer from such label noise, whereas our model is noise-tolerant and more suitable in weakly supervised settings. Overall BERT-MLP performs the best and we further compare it with ours in more perspectives.

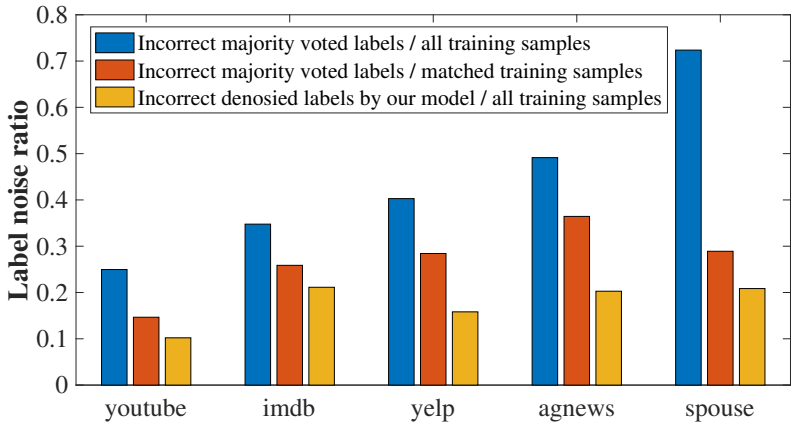


Figure 5.1: The label noise ratio of the initial majority voted labels and our denoised labels in the training set.

5.2.2 Effectiveness of Label Denoising

To study the effectiveness of label denoising, we first compare the label noise ratio in training set given by the majority-voted pseudo labels ($\tilde{Y}$ defined in section 4.2) and our denoised pseudo labels. Figure 5.1 shows that after applying our denoising model, the label noise is reduced by 4.49% (youtube), 4.74% (imdb), 12.6% (yelp), 3.87% (agnews) and 8.06% (spouse) within the matched samples. If we count all the samples, the noise reduction is much more significant with 23.92% by average. Such inspiring results show the effectiveness of our model in denoising weak labels.

Train a Classifier with Denoised Labels We further study how the denoised labels benefit the training of supervised models. To this end, we feed the labels generated by majority voting and denoised ones generated by our model into two state-of-the-art supervised models: ULMFiT and BERT-MLP (described in section 5.1). Table 5.3 shows that denoised labels significantly improve the performance of supervised models on all the datasets.

Table 5.3: Classification accuracy of two supervised methods with labels generated by majority voting and denoised ones generated by our model.

Method	Labels	youtube	imdb	yelp	agnews	spouse
BERT+ MLP	major	77.0	72.5	81.5	75.8	70.7
	ours	**89.8**	**80.2**	**85.8**	**84.3**	**78.0**
UlmFit	major	56.1	70.5	67.3	66.8	72.4
	ours	**90.8**	**81.6**	**85.9**	**84.7**	**81.3**

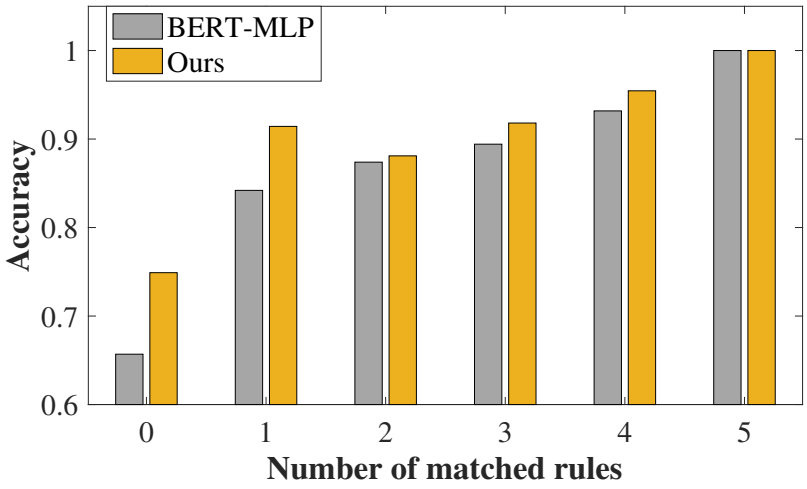


Figure 5.2: Accuracy on low-resource samples (matched by a small number of rules) in Youtube dataset.

5.2.3 Effectiveness of Handling Rule Coverage

We proceed to study how effective our model is when dealing with the low-coverage issue of weak supervision. To this end, we evaluate the performance of our model for the samples covered by different numbers of rules. As shown in Figure 5.2, the strongest baseline (BERT-MLP) trained with majority-voted labels performs poorly on samples that are matched by few rules or even no rules. In contrast, after applying our model, the performance on those less matched samples improves significantly. This is due to the neural classifier in our model, which predicts soft labels for unmatched samples and utilizes the information from the multiple sources through co-training.

5.2.4 Incorporating Clean Labels

We also study how our model can further benefit from a small amount of labeled data. While our model uses weak labels by default, it can easily incorporate clean labeled data by changing the weak labels to clean ones and fix them during training. We study the performance of our model in this setting, and compare with the fully-supervised BERT-MLP model trained with the same amount of clean labeled data.

Table 5.4: The classification accuracy of BERT-MLP and our model with ground truth labeled data

Labeled	Method	youtube	imdb	yelp	agnews	spouse
0.5%	Bert-MLP	80.6	76.9	86.2	82.6	68.2
	Ours	**92.4**	**81.9**	**87.5**	**86.4**	**81.3**
2%	Bert-MLP	83.2	78.8	87.4	84.7	72.3
	Ours	**92.9**	**83.1**	**87.6**	**85.7**	**81.3**
5%	Bert-MLP	87.7	83.6	89.0	86.4	74.8
	Ours	**93.8**	**86.1**	**90.4**	**88.2**	**82.1**
20%	Bert-MLP	90.8	86.0	90.3	**89.2**	75.6
	Ours	**94.0**	**86.1**	**90.5**	**89.2**	**84.5**
50%	Bert-MLP	91.8	**86.2**	**90.5**	89.2	78.0
	Ours	**95.4**	**86.2**	**90.5**	**89.3**	**85.9**
100%	Bert-MLP	94.4	87.2	91.1	90.7	79.6

As shown in Table 5.4, the results of combining our denoised labels with a small amount of clean labels are inspiring: it further improves the performance of our model and consistently outperforms the fully supervised BERT-MLP model. When the labeled ratio is small, the performance improvement over the fully-supervised model is particularly large: improving the accuracy by 6.28% with 0.5% clean labels and 3.84% with 5% clean labels on average. When the ratio of clean labels is large, the performance improvements gradually becomes marginal.

The performance improvement over the fully-supervised model is relatively smaller on yelp and agnews datasets. The reason is likely that the text genres of yelp and agnews are

similar to the text corpora used in BERT pre-training, making the supervised model fast achieve its peak performance with a small amount of labeled data.

5.2.5 Comparison with Supervised Models

To better understand the efficacy of our model, we compare it with supervised text classification models and explore how our denoised labels can benefit these models.

We compare our model's performance with these supervised models when we vary the size of clean labeled data. Note that the clean labeled data are only used for training the two supervised models, whereas our model is still trained by weak supervision.

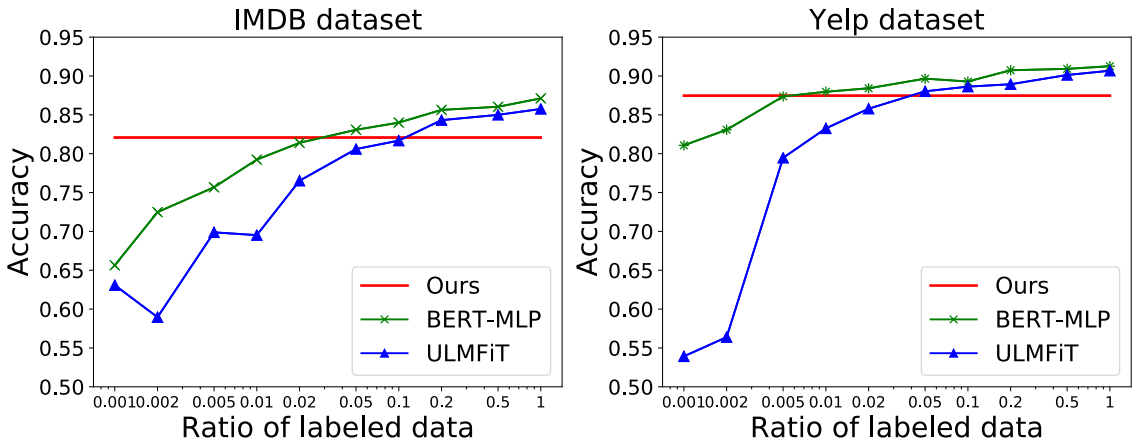


Figure 5.3: Comparison with two fully-supervised models trained by clean labels on imdb and yelp as the ratio of labeled data changes. The blue and green lines indicate classification accuracy of fully-supervised models trained by different percentages of clean labels. The red straight line is classification accuracy of our model trained without clean labels.

Figure 5.3 shows their performance as the ratio of clean labeled data changes on the imdb and yelp dataset. These supervised models, even with pre-trained representations, require a large number of labeled data to achieve the same performance with our model. In contrast, our model largely reduces labeling effort as it needs no annotated data but only needs heuristic rules; this advantage makes our model favorable in applications where labeled data are lacking.

5.2.6 Ablation Study

We perform ablation studies to evaluate the effectiveness of the three components in our model: the label denoiser, the neural classifier, and the self-training over unmatched samples. By removing one of them, we obtain four settings: 1) Rule-only, represents w/o neural classifier and self-training; 2) Neural-only, represents w/o label denoiser and self-training; 3) Neural-self: represents w/o label denoiser; 4) Rule-Neural: represents w/o self training. 3) and 4) are supervised by the initial simple majority voted labels. Table 5.5 shows the results. We find that all the three components are key to our model, because: 1) the rule-based label denoiser iteratively obtains higher-quality pseduo labels from the weak supervision sources; 2) the neural classifier extracts extra supervision signals from unlabeled data through self-training.

Table 5.5: Ablation Study Results.

Method	**youtube**	**imdb**	**yelp**	**agnews**	**spouse**
Ours	**94.9**	**82.9**	**87.5**	**85.7**	**81.3**
Rule-only	90.3	73.1	70.2	63.6	77.2
Neural-only	77.0	72.5	81.5	75.8	70.7
Neural-self	89.3	81.4	82.9	81.3	79.7
Rule-Neural	87.2	80.1	80.8	84.8	69.9

5.2.7 Parameter Study

The primary parameters of our model include: 1) the dimension of hidden layers d_{h} in the label denoiser and the feature-based classifier; 2) learning rate lr; 3) the weight c_1, c_2, and c_3 of regularization term for ℓ_1, ℓ_2, and ℓ_3 in (Equation 4.8); 4) We fix momentum term $\alpha = 0.6$ followed the implementation of [58]. By default, we set $d_{\mathrm{h}} = 128$, $lr = 0.02$, and $c_1 = 0.2, c_2 = 0.7, c_3 = 0.1$ as our model achieves overall good performance with these parameters. The search space of d_{h} is 2^{6-9}, lr is $0.01 - 0.1$, c_1 and c_3 are $0.1 - 0.9$ (note that $c_2 = 1 - c_1 - c_3$). The hyperparameter configuration for the best performance reported in Table 5.2 is shown in the section B.2.

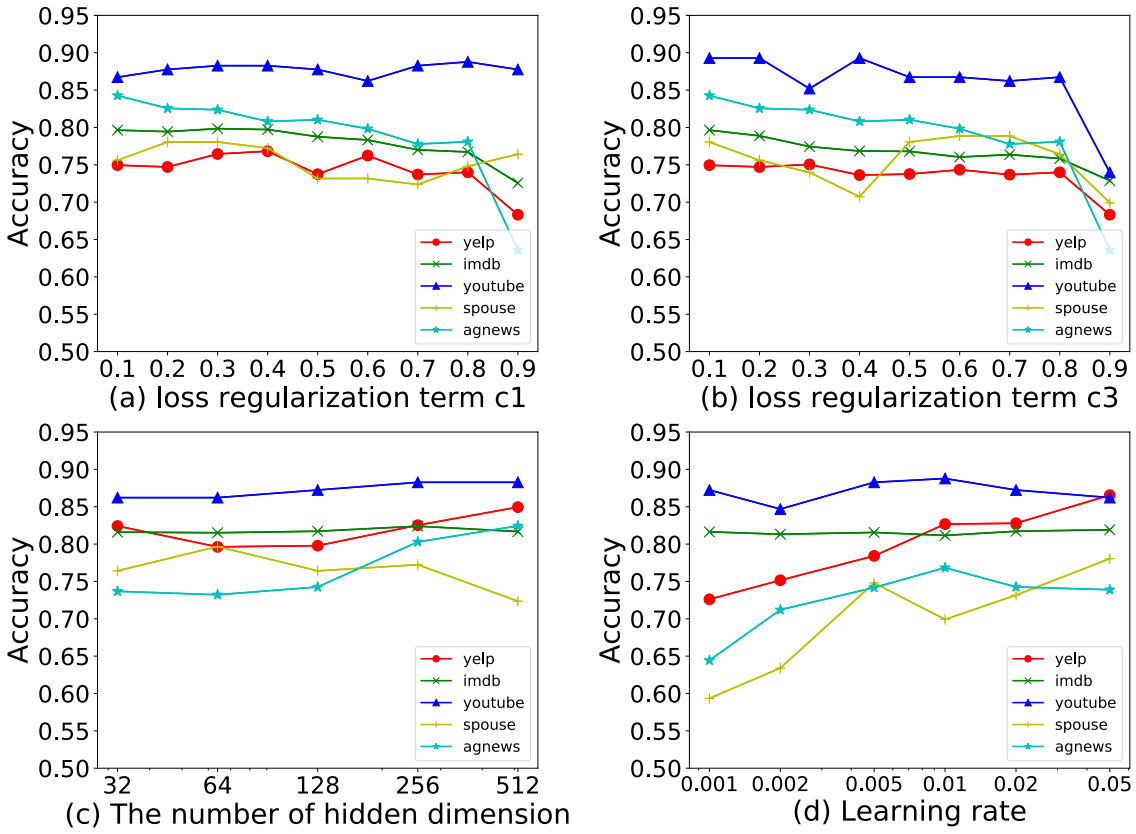


Figure 5.4: The prediction accuracy over different parameter settings.

We test the effect of one hyperparameter by fixing others to their default values. In Figure 5.4 (a) and (b), we find the performance is stable except that the loss weight is too large. For (c) and (d), except for the spouse dataset when lr is too small and d_{h} is too large (instability due to the dataset size is small), our model is robust to the hyperparameters when they are in a reasonable range. We also report overall performance for all the search trails in Table B.5 of section B.2.

CHAPTER 6
CONCLUSION

We have proposed a deep neural text classifier learned not from excessive labeled data, but only unlabeled data plus weak supervisions, in an end-to-end manner. The major challenges of learning from weak supervision are the label noise issue and the low-coverage issue. Our model addresses these two challenges using two components that co-train each other: (1) a rule-based classifier that estimates source reliability to reduce label noise on the matched samples, (2) a neural classifier that learns distributed representations and predicts over all the samples. The two components are integrated into a co-training framework to benefit from each other and can be trained end-to-end. In our experiments, we find our model effectively tackles the two challenges in weakly supervised text classification. It not only outperforms state-of-the-art weakly supervised models, but also benefits supervised models with its denoised labeled data. Our model makes it possible to train accurate deep text classifiers using easy-to-provide rules. It can thus largely reduce human labeling efforts for deep text classification and will be highly suitable in applications where labeled data are expensive to obtain. As future work, we are interested in thoroughly integrating our model with pre-trained language models to improve its performance further, as well as extending the co-training framework to other predictive tasks beyond text classification.

Appendices

APPENDIX A

DATA PROCESSING

Data was processed before being added to this document.

A.1 Dataset Preparation

We randomly split the full datasets into three parts – a training set, a validation set and a test set, with ratios of 80%, 10% and 10%, respectively. The splitting is fixed for all the methods for fair comparisons. We use the training set to train the model, the validation set to for optimal early stopping and hyperparameters fine-tuning, and finally evaluate different methods on the test set.

Recall our definition of the matched corpus $\mathcal{D}_L$. In practice, we only regard instances covered by more than p sources as "matched" instances, where $p \in [0, 1, 2, \ldots k-1]$. Specifically, p is set to $2, 1, 1, 0, 0$ for YouTube, Yelp, IMDB, AGNews, and Spouse datasets.

We obtain the pre-trained BERT embeddings from the 'bert-base-uncased' model. Our pre-processed data with the BERT embeddings and weak labels are available to download at our Google Drive. The dataset description can be found in our Github repo

A.2 Labeling Sources

We have four types of annotation rules which are Keyword Labeling Sources, Pattern-matching (Regular Expressions) Labeling Sources, Heuristic Labeling Sources, and Third-party Tools. For the first and second one, we give the uniform definitions for all the datasets.

- *Keyword Labeling Sources*

  Given x as a document $\boldsymbol{d_i}$ in a corpus of text documents $\mathcal{D}$, a keywords list L, and a class label C in the set of target classes $\mathcal{C}$, we define keywords matching annotation

process `HAS` as

Definition 2 (Keywords rules)**.** `HAS(x, L)` $\Rightarrow$ `C` if x matches one of the words in the list L.

- *Pattern-matching Labeling Sources*

  Given x, a regular expression R, and a class label C, we define the pattern-matching annotation process `MATCH` as

  Definition 3 (Pattern-matching rules)**.** `MATCH(x, R)` $\Rightarrow$ `C` if x matches the regular expression R.

For the remaining third and fourth types, each dataset has specific definitions. We then state all the labeling rules for each dataset from Table A.2 to Table A.6.

A.2.1 Statistics of Labeling Sources

We show the accuracy and coverage of each rule in the Figure A.1, where the shape represents the coverage and the color depth represents the accuracy of the rule-induced labeled data. The average accuracy of these rules is 67.5%, and the average coverage is 23.3%.

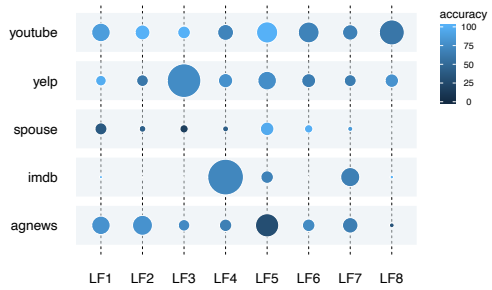


Figure A.1: The coverage and accuracy of our used labeling functions on five datasets. Larger circle denotes higher coverage and lighter color denotes higher accuracy.

We also show one example of Yelp dataset with the detail statistics for each labeling source, and the rule descriptions are in Table A.4.

Table A.1: The labeling rules statictics for Yelp dataset. Both Coverage and Emp. Accu (number of corrected samples / rule-matched samples) are in %.

Labeling source	Coverage	Emp. Accu
textblob	6.80	97.06
keyword_recommand	8.40	59.52
keyword_general	75.20	74.20
keyword_mood	12.80	78.12
keyword_service	33.30	75.68
keyword_price	23.30	63.93
keyword_environment	8.80	63.64
keyword_food	11.40	78.95

A.2.2 Rules Description

We show some examples of labeling rules here, and the full description of rules and their corresponding weak labels are in our Github repo [1].

Youtube We use the same labeling functions as ratner2017snorkel, and we show the rules with an example in Table A.2.

IMDB The rules are straightforward so we show the rules without the sentence examples in Table A.3.

Yelp The rules are straightforward so we show the rules without the sentence examples in Table A.4. We provide labeling rules in eight views.

AGnews The rules are straightforward so we show the rules without the sentence examples in Table A.5.

Spouse We use the same rule as ratner2017snorkel and we show the definition as well as examples in Table A.6.

[1] https://github.com/weakrules/Denoise-multi-weak-sources/tree/master/rules-noisy-labels

Table A.2: Youtube labeling sources examples

Rule	**Example**
`HAS(x,[my]) ⇒ SPAM`	Plizz withing my channel
`HAS(x, [subscribe]) ⇒ SPAM`	Subscribe to me and I'll subscribe back!!
`HAS(x, [http]) ⇒ SPAM`	please like : http://www.bubblews.com/news/9277547-peace-and-brotherhood
`HAS(x, [please, plz]) ⇒ SPAM`	Please help me go here http://www.gofundme.com/littlebrother
`HAS(x, [song]) ⇒ HAM`	This song is great there are 2,127,315,950 views wow
`MATCH(x, check.*out ) ⇒ SPAM`	Please check out my vidios
We define `LENGTH(x)` as the number of words in x. `LENGTH(x) < 5 ⇒ HAM`	2 BILLION!!
We define $x.ents$ as the tokens of x, and $x.ent.label$ as its label. $LENGTH(x) < 20 \ AND \ any([ent.label == PERSON \ for \ ent \ in \ x.ents] \Rightarrow HAM$	Katy Perry is garbage. Rihanna is the best singer in the world.
We define `POLARITY(x)` as the sentiment subjectivity score obtained from the TextBlob tool, a pretrained sentiment analyzer. $POLARITY(x) > 0.9 \Rightarrow HAM$	Discover a beautiful song of A young Moroccan http://www.linkbucks.com/AcN2g

Table A.3: IMDB labeling sources examples

Rule
[masterpiece, outstanding, perfect, great, good, nice, best, excellent, worthy, awesome, enjoy, positive, pleasant, wonderful, amazing, superb, fantastic, marvellous, fabulous] ⇒ POS
[bad, worst, horrible, awful, terrible, crap, shit, garbage, rubbish, waste] ⇒ NEG
[beautiful, handsome, talented]⇒ POS
[fast forward, n t finish] ⇒ NEG
[well written, absorbing, attractive, innovative, instructive, interesting, touching, moving]⇒ POS
[to sleep, fell asleep, boring, dull, plain]⇒ NEG
[than this, than the film, than the movie]⇒ NEG
MATCH(x, *PRE*EXP*) ⇒ POS PRE = [will, ll , would , d , can t wait to] EXP = [next time, again, rewatch, anymore, rewind]
MATCH(x, *PRE*EXP*) ⇒ POS PRE = [highly, do, would, definitely, certainly, strongly, i, we] EXP = [recommend, nominate]
MATCH(x, *PRE*EXP*) ⇒ POS PRE = [high, timeless, priceless, has, great, real, instructive] EXP = [value, quality, meaning, significance]

Table A.4: Yelp labeling sources examples

View	Rule
General	[outstanding, perfect, great, good, nice, best, excellent, worthy, awesome, enjoy, positive, pleasant,wonderful, amazing] ⇒ POS
General	[bad, worst, horrible, awful, terrible, nasty, shit, distasteful,dreadful, negative]⇒ NEG
Mood	[happy, pleased, delighted,contented, glad, thankful, satisfied] ⇒ POS
Mood	[sad, annoy, disappointed,frustrated, upset, irritated, harassed, angry, pissed]⇒ NEG
Service	[friendly, patient, considerate, enthusiastic, attentive, thoughtful, kind, caring, helpful, polite, efficient, prompt] ⇒ POS
Service	[slow, offended, rude, indifferent, arrogant]⇒ NEG
Price	[cheap, reasonable, inexpensive, economical] ⇒ POS
Price	[overpriced, expensive, costly, high-priced]⇒ NEG
Environment	[clean, neat, quiet, comfortable, convenien, tidy, orderly, cosy, homely] ⇒ POS
Environment	[noisy, mess, chaos, dirty, foul]⇒ NEG
Food	[tasty, yummy, delicious,appetizing, good-tasting, delectable, savoury, luscious, palatable] ⇒ POS
Food	[disgusting, gross, insipid]⇒ NEG
	[recommend] ⇒ POS
Third-party Tools	$POLARITY(x) > 0.5 \Rightarrow$ **POS** $POLARITY(x) > 0.5 \Rightarrow$ **NEG**

Table A.5: AGnews labeling sources examples

Rule
`[ war , prime minister, president, commander, minister, annan, military, militant, kill, operator] ⇒ POLITICS`
`[baseball, basketball, soccer, football, boxing, swimming, world cup, nba,olympics,final, fifa] ⇒ SPORTS`
`[delta, cola, toyota, costco, gucci, citibank, airlines] ⇒ BUSINESS`
`[technology, engineering, science, research, cpu, windows, unix, system, computing, compute] ⇒ TECHNOLOGY`

Table A.6: Spouse labeling sources examples

Rule	**Example**
`[father, mother, sister, brother, son, daughter, grandfather, grandmother, uncle, aunt, cousin] ⇒ NEG`	His 'exaggerated' sob stories allegedly include claiming he had cancer, and that his son had made a suicide attempt.
`[boyfriend, girlfriend, boss, employee, secretary, co-worker] ⇒ NEG`	Dawn Airey's departure as European boss of Yahoo after just two years will bring a smile to the face of Armando Iannucci.
`MATCH(x, *PERSON1*LIST*PERSON2* ⇒ POS LIST = [spouse, wife, husband, ex-wife, ex-husband]`	On their wedding day, last week sundayGhanaian actress Rose Mensah, popularly known as Kyeiwaa, has divorced her husband Daniel Osei, less than four days after the glamorous event.
We define `LASTNAME(x)` as the last name of x. `LASTNAME(person1) == LASTNAME(person2) ⇒ POS`	Karen Bruk and Steven Bruk, Mrs. Bruk's spouse, exercise shared investment power over the Shares of the Company held by Karen Bruk and KMB.

APPENDIX B

EXPERIMENT EXPLANATION

B.1 Model Training

Computing infrastructure Our code can be run on either CPU or GPU environment with Python 3.6 and Pytorch.

Running time Our model consists of two simple MLP networks with **0.199M** trainable parameters, thus the model is very time efficient with the avearge running time **4.5 minutes**. The running time differ based on the dataset size. We test our code on the System Ubuntu 18.04.4 LTS with *CPU: Intel(R) Xeon(R) Silver 4214 CPU @ 2.20GHz* and *GPU: NVIDIA GeForce RTX 2080*. All the models are trained for a maximum of 500 epochs.

Validation performance For the main results in Table 5.2, the corresponding validation accuracy for our model is shown in Table B.2.

B.2 Hyperparameter Search

Since our datasets are well balanced, we use *accuracy* as the criterion for optimal early stopping and hyperparameters fine-tuning. Our hyperparameter values are uniform sampled within a reasonable range with particular numbers in Table B.3.

Table B.4 shows the hyper parameters used to get the best results for Table 5.2.

Table B.1: Running time for one experiment on CPU for five datasets in minutes

Dataset	**youtube**	**imdb**	**yelp**	**agnews**	**spouse**
Running time (min)	1.9	3.65	3.92	11.92	1.5

Table B.2: validation accuracy on for five datasets of the main results in Table 5.2.

Dataset	**youtube**	**imdb**	**yelp**	**agnews**	**spouse**
Validation accuracy	87.8	81.8	88.2	85.6	79.7
Test accuracy	94.9	82.9	87.5	85.7	81.3

Table B.3: The hyper parameters search bounds.

Parameters	**Search Range**
d_{h}	32, 64, 128, 256, 512
lr	0.001, 0.002, 0.005, 0.01, 0.02, 0.05
c_1	0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9
c_3	0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9

Table B.4: The hyper parameters setting for the best accuracy results of Table 5.2.

Parameters	**youtube**	**imdb**	**yelp**	**agnews**	**spouse**
d_{h}	128	64	128	256	256
lr	0.02	0.02	0.02	0.05	0.02
c_1	0.2	0.2	0.2	0.1	0.2
c_3	0.1	0.2	0.2	0.1	0.1

Table B.5: The validation and test results for the hyperparameters search trails with the mean and standard deviation.

	youtube	**imdb**	**yelp**	**agnews**	**spouse**
Val Mean	81.5	77.1	79.1	80.0	83.5
Val Stdev	0.019	0.036	0.034	0.073	0.093
Test Mean	87.1	78.0	77.2	79.8	79.5
Test Stdev	0.021	0.031	0.042	0.070	0.118

For the above four parameters with their range, we perform 1350 search trails. The test and validation results accuracy with mean and standard deviation for hyperparameters search experiments are in Table B.5.

REFERENCES

[1] A. Gandomi and M. Haider, "Beyond the hype: Big data concepts, methods, and analytics," *International journal of information management*, vol. 35, no. 2, pp. 137–144, 2015.

[2] L. Deng and Y. Liu, *Deep learning in natural language processing*. Springer, 2018.

[3] P. Badjatiya, S. Gupta, M. Gupta, and V. Varma, "Deep learning for hate speech detection in tweets," in *Proceedings of the 26th International Conference on World Wide Web Companion*, ser. WWW '17 Companion, Perth, Australia: International World Wide Web Conferences Steering Committee, 2017, pp. 759–760, ISBN: 9781450349147.

[4] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds., Curran Associates, Inc., 2015, pp. 649–657.

[5] A. Krebs, A. Lenci, and D. Paperno, "SemEval-2018 task 10: Capturing discriminative attributes," in *Proceedings of The 12th International Workshop on Semantic Evaluation*, New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 732–740.

[6] M. T. Pilehvar and J. Camacho-Collados, "Wic: The word-in-context dataset for evaluating context-sensitive meaning representations," *arXiv preprint arXiv:1808.09121*, 2018.

[7] Y. Kim, "Convolutional neural networks for sentence classification," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1746–1751.

[8] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, "A convolutional neural network for modelling sentences," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Baltimore, Maryland: Association for Computational Linguistics, Jun. 2014, pp. 655–665.

[9] S. Lai, L. Xu, K. Liu, and J. Zhao, "Recurrent convolutional neural networks for text classification," in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, ser. AAAI'15, Austin, Texas: AAAI Press, 2015, pp. 2267–2273, ISBN: 0262511290.

[10] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019*

Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186.

[11] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. Le, and R. Salakhutdinov, "Transformer-XL: Attentive language models beyond a fixed-length context," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 2978–2988.

[12] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, *Roberta: A robustly optimized bert pretraining approach*, 2019. arXiv: 1907.11692 [cs.CL].

[13] A. Ratner, S. H. Bach, H. Ehrenberg, J. Fries, S. Wu, and C. Ré, "Snorkel: Rapid training data creation with weak supervision," *Proc. VLDB Endow.*, vol. 11, no. 3, pp. 269–282, Nov. 2017.

[14] Y. Meng, J. Shen, C. Zhang, and J. Han, "Weakly-supervised neural text classification," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, ACM, 2018, pp. 983–992.

[15] H. Zamani, W. B. Croft, and J. S. Culpepper, "Neural query performance prediction using weak supervision from multiple signals," in *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, 2018, pp. 105–114.

[16] A. Awasthi, S. Ghosh, R. Goyal, and S. Sarawagi, "Learning from rules generalizing labeled exemplars," in *International Conference on Learning Representations*, 2020.

[17] D. Bahdanau, K. Cho, and Y. Bengio, *Neural machine translation by jointly learning to align and translate*, cite arxiv:1409.0473 Comment: Accepted at ICLR 2015 as oral presentation, 2014.

[18] A. Tarvainen and H. Valpola, "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., Curran Associates, Inc., 2017, pp. 1195–1204.

[19] J. Howard and S. Ruder, "Universal language model fine-tuning for text classification," *arXiv preprint arXiv:1801.06146*, 2018.

[20] K. Kowsari, K. Jafari Meimandi, M. Heidarysafa, S. Mendu, L. Barnes, and D. Brown, "Text classification algorithms: A survey," *Information*, vol. 10, no. 4, p. 150, 2019.

[21] G. Lebanon, Y. Mao, and J. Dillon, "The locally weighted bag of words framework for document representation," *Journal of Machine Learning Research*, vol. 8, no. Oct, pp. 2405–2441, 2007.

[22] J. Ramos *et al.*, "Using tf-idf to determine word relevance in document queries," in *Proceedings of the first instructional conference on machine learning*, New Jersey, USA, vol. 242, 2003, pp. 133–142.

[23] J. Liu, J. T. Wang, W. Hsu, and K. G. Herbert, "Xml clustering by principal component analysis," in *16th IEEE International Conference on Tools with Artificial Intelligence*, IEEE, 2004, pp. 658–662.

[24] S. Balakrishnama and A. Ganapathiraju, "Linear discriminant analysis-a brief tutorial," in *Institute for Signal and information Processing*, vol. 18, 1998, pp. 1–8.

[25] L. Li, C. R. Weinberg, T. A. Darden, and L. G. Pedersen, "Gene selection for sample classification based on gene expression data: Study of sensitivity to choice of parameters of the ga/knn method," *Bioinformatics*, vol. 17, no. 12, pp. 1131–1142, 2001.

[26] L. M. Manevitz and M. Yousef, "One-class svms for document classification," *Journal of machine Learning research*, vol. 2, no. Dec, pp. 139–154, 2001.

[27] B. Xu, X. Guo, Y. Ye, and J. Cheng, "An improved random forest classifier for text categorization.," *JCP*, vol. 7, no. 12, pp. 2913–2920, 2012.

[28] U. Cambridge, "Introduction to information retrieval," 2009.

[29] J. Huang and C. X. Ling, "Using auc and accuracy in evaluating learning algorithms," *IEEE Transactions on knowledge and Data Engineering*, vol. 17, no. 3, pp. 299–310, 2005.

[30] J. A. Hanley and B. J. McNeil, "The meaning and use of the area under a receiver operating characteristic (roc) curve.," *Radiology*, vol. 143, no. 1, pp. 29–36, 1982.

[31] B. W. Matthews, "Comparison of the predicted and observed secondary structure of t4 phage lysozyme," *Biochimica et Biophysica Acta (BBA)-Protein Structure*, vol. 405, no. 2, pp. 442–451, 1975.

[32] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[33] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[34] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, 2017.

[35] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[36] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, "Hierarchical attention networks for document classification," in *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, 2016, pp. 1480–1489.

[37] X. Qiu, T. Sun, Y. Xu, Y. Shao, N. Dai, and X. Huang, "Pre-trained models for natural language processing: A survey," *arXiv preprint arXiv:2003.08271*, 2020.

[38] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," *arXiv preprint arXiv:1802.05365*, 2018.

[39] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, *Improving language understanding by generative pre-training*, 2018.

[40] M. Joglekar, H. Garcia-Molina, and A. G. Parameswaran, "Comprehensive and reliable crowd assessment algorithms," in *Proceedings of the IEEE International Conference on Data Engineering*, 2015, pp. 195–206.

[41] N. B. Shah, D. Zhou, and Y. Peres, "Approval voting and incentives in crowdsourcing," in *Proceedings of the International Conference on Machine Learning*, vol. 37, 2015, pp. 10–19.

[42] A. J. Ratner, C. D. Sa, S. W. 0002, D. Selsam, and C. Ré, "Data programming - creating large training sets, quickly," in *Proceedings of the Annual Conference on Neural Information Processing Systems*, 2016.

[43] A. Ratner, B. Hancock, J. Dunnmon, F. Sala, S. Pandey, and C. Ré, "Training complex models with multi-task weak supervision," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019, pp. 4763–4771.

[44] C. E. Brodley and M. A. Friedl, "Identifying mislabeled training data," *Journal of Artificial Intelligence Research*, vol. 11, pp. 131–167, 1999.

[45] M. R. Smith and T. Martinez, "Improving classification accuracy by identifying and removing instances that should be misclassified," in *International Joint Conference on Neural Networks*, 2011.

[46] Y. Yang, W. Chen, Z. Li, Z. He, and M. Zhang, "Distantly supervised ner with partial annotation learning and reinforcement learning," in *Proceedings of the International Conference on Computational Linguistics*, 2018, pp. 2159–2169.

[47] S. Sukhbaatar and R. Fergus, "Learning from noisy labels with deep neural networks," in *Proceedings of the International Conference on Learning Representations*, 2014.

[48] S. Sukhbaatar, J. Bruna, M. Paluri, L. Bourdev, and R. Fergus, "Training convolutional networks with noisy labels," *arXiv:1406.2080 [cs]*, 2014.

[49] J. Goldberger and E. Ben-Reuven, "Training deep neural-networks using a noise adaptation layer," 2016.

[50] M. Dehghani, A. Severyn, S. Rothe, and J. Kamps, "Avoiding your teacher's mistakes: Training neural networks with controlled weak supervision," *CoRR*, vol. abs/1711.00313, 2017.

[51] Z.-Y. Zhang, P. Zhao, Y. Jiang, and Z.-H. Zhou, "Learning from incomplete and inaccurate supervision," in *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2019, pp. 1017–1025.

[52] A. Veit, N. Alldrin, G. Chechik, I. Krasin, A. Gupta, and S. Belongie, "Learning from noisy large-scale datasets with minimal supervision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 6575–6583.

[53] M. Hu, H. Han, S. Shan, and X. Chen, "Weakly supervised image classification through noise regularization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 11 517–11 525.

[54] D. Yarowsky, "Unsupervised word sense disambiguation rivaling supervised methods," in *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 1995, pp. 189–196.

[55] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *Proceedings of the International Conference on Machine Learning*, 2017.

[56] A. Blum and T. Mitchell, "Combining labeled and unlabeled data with co-training," in *Proceedings of the Annual Conference on Computational Learning Theory*, 1998, p. 10.

[57] S. Qiao, W. Shen, Z. Zhang, B. Wang, and A. L. Yuille, "Deep co-training for semi-supervised image recognition," in *Proceedings of the European Conference on Computer Vision*, 2018, pp. 142–159.

[58] S. Laine and T. Aila, "Temporal ensembling for semi-supervised learning," *arXiv preprint arXiv:1610.02242*, 2016.

[59] D. Chen, W. Wang, W. Gao, and Z. Zhou, "Tri-net for semi-supervised deep learning," in *Proceedings of the International Joint Conference on Artificial Intelligence*, 2018, pp. 2014–2020.

[60] T. C. Alberto, J. V. Lochter, and T. A. Almeida, "Tubespam: Comment spam filtering on youtube," in *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, IEEE, 2015, pp. 138–143.

[61] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1*, Association for Computational Linguistics, 2011, pp. 142–150.